FAS solution:

Bài toán qui về với mỗi đỉnh u của cây, tính tổng bình phương các khoảng cách từ u đến các đỉnh còn lại.

1) Đặt s[u] là số lượng đỉnh trong cây con gốc u ta có công thức qui hoạch động:

$$s[u] = 1 + \sum\{s[v]: Prev[v] = u\}$$

2) Đặt $t_1[u]$ là tổng độ dài từ các đỉnh thuộc cây con gốc u đến u. Gọi v là một đỉnh con của u. Khi đó độ dài con đường từ một đỉnh thuộc cây con gốc v đến đỉnh u luôn có dạng $(x + L_{uv})$ với x là độ dài từ đỉnh này đến v. Lấy tổng các biểu thức dạng trên ta thu được $t_1[v] + L_{uv} \cdot s[v]$

Vậy nên:

$$t_1[u] = \sum_{Prev[v]=u}(t_1[v] + L_{uv} \cdot s[v])$$

3) Đặt $t_2[u]$ là tổng độ dài từ các đỉnh không thuộc cây con gốc u đến u. Gọi $w = Prev[u]$ ta có các đỉnh trong trường hợp này được chia thành hai loại:

+) Các đỉnh không thuộc cây con gốc w. Độ dài của nó có dạng $(x + L_{wu})$ với x là độ dài từ đỉnh này đến w. Lấy tổng các biểu thức dạng trên ta thu được $t_2[w] + L_{wu} \cdot (n - s[w])$

+) Các đỉnh thuộc cây con gốc w nhưng không thuộc cây con gốc u. Độ dài của nó có dạng $(y + L_{wu})$ với y là độ dài đến w. Lấy tổng ta thu được $\sum y + L_{wu} \cdot \sum 1$

*) $\sum y$ = tổng độ dài từ các đỉnh trong cây con gốc w đến w tbỏ đi các đỉnh thuộc nhánh u. Ta có:

$$\sum y = t_1[w] - (t_1[u] + L_{wu} \cdot s[u])$$

*) $\sum 1$ = số lượng đỉnh trong cây con gốc w trừ đi số lượng đỉnh trong cây con gốc u. Ta có:

$\sum 1 = s[w] - s[u]$

$\sum (L+x)^2 = L^2 \sum 1 + 2L \sum x + \sum x^2$

Vậy nên:

$t_2[u] = t_2[w] + L_{wu} \cdot (n - s[w]) + t_1[w] - (t_1[u] + L_{wu} \cdot s[u]) + L_{wu} \cdot (s[w] - s[u])$

$= t_2[w] + t_1[w] - t_1[u] + L_{wu} \cdot (n - 2 \cdot s[u])$

4) Đặt $f_1[u]$ là tổng bình phương độ dài từ các đỉnh thuộc cây con gốc u đến u. Với đỉnh v là con của đỉnh u thì bình phương độ dài luôn có dạng $(x + L_{uv})^2 = x^2 + 2L_{uv} \cdot x + L_{uv}^2$. Ở đây x là độ dài từ đỉnh này đến v. Do đó ta có:

$$f_1[u] = \sum_{Prev[v]=u}(f_1[v] + 2L_{uv} \cdot t_1[v] + L_{uv}^2 \cdot s[v])$$

5) Đặt $f_2[u]$ là tổng bình phương độ dài từ các đỉnh không thuộc cây con gốc u đến u.

Đặt $w = Prev[u]$. Các đỉnh thuộc loại này được chia thành hai loại

+) Các đỉnh không thuộc cây con gốc w. Bình phương độ dài của nó có dạng $(x + L_{wu})^2 = x^2 + 2L_{wu} \cdot x + L_{wu}^2$ với x là độ dài đến đỉnh w. Lấy tổng các biểu thức trên thu được:

$$f_2[w] + 2L_{wu} \cdot t_2[w] + L_{wu}^2 \cdot (n - s[w])$$

+) Các đỉnh thuộc cây con gốc w nhưng không thuộc cây con gốc u. Bình phương độ dài của nó có dạng $(y + L_{wu})^2 = y^2 + 2L_{wu} \cdot y + L_{wu}^2$. Lấy tổng ta có $\sum y^2 + 2L_{wu} \cdot \sum y + L_{wu}^2 \cdot \sum 1$

Cần tính thêm:

$\sum y^2$ = tổng bình phương độ dài từ các đỉnh trong cây con gốc w đến w trừ đi tổng bình phương độ dài từ các đỉnh trong cây con gốc u. Ta có:

$$\sum y^2 = f_1[w] - (f_1[u] + 2L_{wu} \cdot t_1[u] + L_{wu}^2 \cdot s[u])$$

Từ các công thức dẫn xất ở trên ta có:

$f_2[u] = f_2[w] + 2L_{wu} \cdot t_2[w] + L_{wu}^2 \cdot (n - s[w]) + f_1[w] - (f_1[u] + 2L_{wu} \cdot t_1[u] + L_{wu}^2 \cdot s[u])$

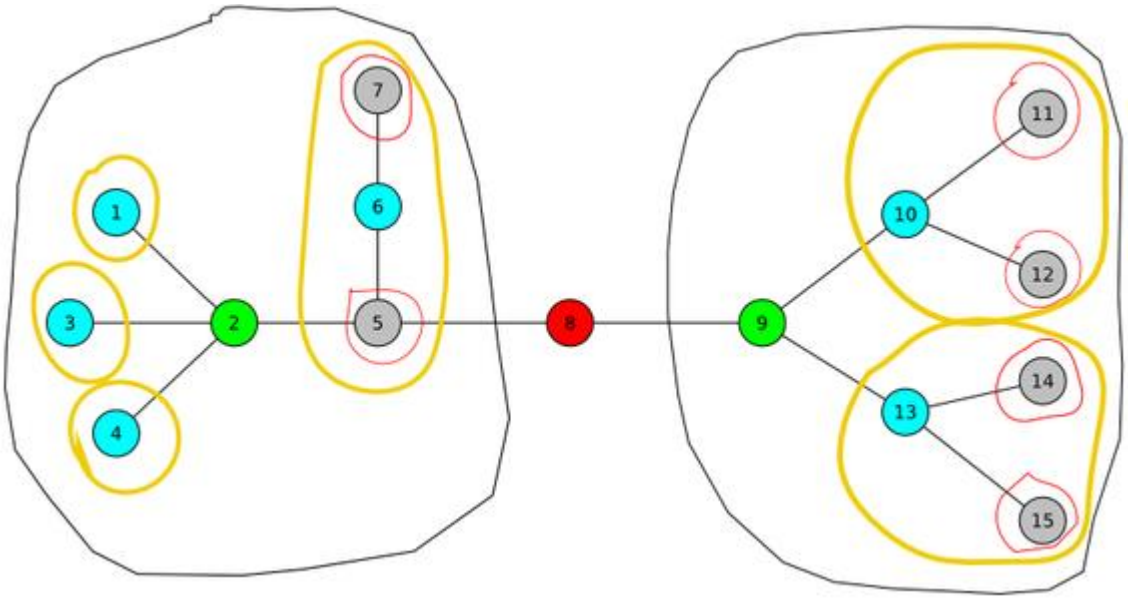
$+2L_{wu} \cdot (t_1[w] - t_1[u] - L_{wu} \cdot s[u]) + L_{wu}^2 \cdot (s[w] - s[u])$

Để tính các mảng s, t_1, f_1 thực hiện qui hoạch động ngược từ cuối sắp xếp topo

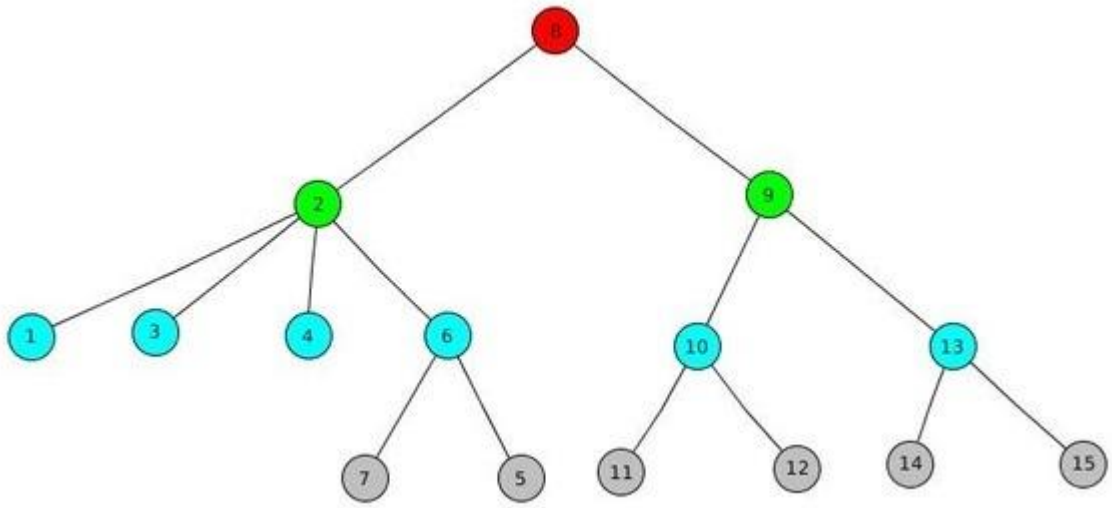
Để tính các mảng t_2, f_2 thực hiện qui hoạch động xuôi.

Chú ý kiểu kết quả là số nguyên 64 bits

Ta sẽ xây dựng mô hình đệ quy tạo ra một cây centroid từ cây ban đầu như sau. Lấy ra một centroid ban đầu, cây sẽ phân rã thành nhiều cây khác nhau mà mỗi cây sẽ không có quá $\left\lfloor \frac{N}{2} \right\rfloor$ đỉnh so với cây gốc. Dùng centroid này làm gốc của cây centroid, gọi đệ quy phân rã các cây mới tạo ra, lấy ra những centroid của cây đó làm đỉnh con của centroid trước vừa tìm được. Tiếp tục thực hiện thuật toán đệ quy cho tới khi phân rã hết mọi cây thành những đỉnh centroid khác nhau.



Hình minh họa 2



Khi thuật toán kết thúc ta sẽ thu được cây centroid như sau:

Hình minh họa 3

3.1 Tính chất

Cây Centroid có những tính chất cơ bản sau đây:

- Cây chứa hết tất cả những đỉnh của cây cũ.
- Cây có độ cao lớn nhất là log(n). Dễ thấy cứ mỗi giai đoạn phân ra, số lượng đỉnh của mỗi cây mới không vượt quá một nửa của cây ban đầu vậy nên độ cao của cây có thể tính bằng câu hỏi đơn giản: "Có thể chia một số N cho 2 bao nhiêu lần thì nó sẽ nhỏ hơn hoặc bằng 1". Hiển nhiên câu trả lời là là $\log_2 N$.
- Với hai đỉnh A, B bất kì từ cây ban đầu, đường đi giữa chúng trên cây gốc đều sẽ trung gian qua một đỉnh C (với C là LCA của A và B ở cây centroid).
- Lưu ý: cây centroid chỉ là "giả", khoảng cách và đường đi giữa hai đỉnh trên này khác với khoảng cách thực của nó ở cây gốc

```
#include <bits/stdc++.h>
#define maxn 200001

using namespace std;

int n, K;
vector<int> adj[maxn];
int Tsize, s[maxn];
int par[maxn];
int nho[maxn], cur[maxn];
long long ans = 0;

void DFS(int u, int p, int L) {
    if (L > K)
```

```
        return;
    ans += nho[K - L];
    ++cur[L];
    for(int v : adj[u])
        if (v != p && !par[v])
            DFS(v, u, L + 1);
}

void ReSubsize(int u, int p) {
    ++Tsize;
    s[u] = 1;
    for(int v : adj[u])
        if (v != p && !par[v]) {
            ReSubsize(v, u);
            s[u] += s[v];
        }
}

int GetCentroid(int u, int p) {
    for(int v : adj[u])
        if (v != p && !par[v] && s[v] > Tsize / 2)
            return GetCentroid(v, u);
    return u;
}

void Descompose(int u, int p) {
    Tsize = 0;
    ReSubsize(u, 0);
    if (Tsize < K + 1)
        return;
    int centroid = GetCentroid(u, 0);
    if (p)
        par[centroid] = p;
    else
        par[centroid] = centroid;

    // Thuc hien dem so cap duong di co k canh
    if (Tsize > K) {
        for(int i = 0; i <= min(Tsize - 1, K); ++i)
            nho[i] = 0;
        nho[0] = 1;
        for(int u : adj[centroid])
            if (!par[u]) {
                for(int i = 0; i <= min(Tsize - 1, K); ++i)
                    cur[i] = 0;
                DFS(u, centroid, 1);
                for(int i = 0; i <= min(Tsize - 1, K); ++i)
                    nho[i] += cur[i];
            }
    }

    // Xoay cay
    for(int v : adj[centroid])
        if (!par[v])
            Descompose(v, centroid);

}

#define task "FIXEDLENGHT1"
int main() {
    if (fopen(task".inp", "r")) {
        freopen(task".inp", "r", stdin);
        freopen(task".out", "w", stdout);
    }
```

```
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    cin >> n >> K;
    for(int i = 1; i < n; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    Descompose(1, 0);


    cout << ans;
}
```

CCOLOR:

```
#include <bits/stdc++.h>
#define maxn 100005

using namespace std;

int n, k;
vector<int> adj[maxn];
int s[maxn], xoa[maxn], Pr[maxn];
int d[maxn], Pa[maxn];
int m = 0;
int start[maxn];
pair<int, int> RMQ[20][2 * maxn];
int color[maxn];
multiset<int> se[maxn];


void DFS (int u, int dad) {
   d[u] = d[dad] + 1;
   start[u] = ++m;
   RMQ[0][m] = {d[u], u};
   for (int v : adj[u])
      if (v != dad) {
         DFS (v, u);
         RMQ[0][++m] = {d[u], u};
      }
}

void BuildRMQ() {
   int limit = log2 (m);
   for (int k = 1; k <= limit; ++k) {
      for (int i = 1; i + (1 << k) - 1 <= m; ++i) {
```

```
            RMQ[k][i] = min (RMQ[k - 1][i], RMQ[k - 1][i + (1 << (k - 1))]);
        }
    }
}

int LCA (int u, int v) {
    int x = start[u], y = start[v];
    if (x > y)
        swap (x, y);
    int k = log2 (y - x + 1);
    return min (RMQ[k][x], RMQ[k][y - (1 << k) + 1]).second;
}

int distance (int u, int v) {
    int w = LCA (u, v);
    return d[u] + d[v] - 2 * d[w];
}

void DFS_size (int u, int dad) {
    s[u] = 1;
    for (int v : adj[u])
        if (v != dad && !xoa[v]) {
            Pr[v] = u;
            DFS_size (v, u);
            s[u] += s[v];
        }
}

void Centroid_decompose (int r, int dad) {
    Pr[r] = 0;
    DFS_size (r, 0);
    int siz = s[r], smax;
    while (1) {
        smax = siz - s[r];
        int u = 0;
        for (int v : adj[r])
            if (!xoa[v]) {
                if (Pr[v] == r && smax < s[v]) {
                    smax = s[v];
                    u = v;
                }
            }
        if (smax <= siz / 2)
            break;
        r = u;
    }
```

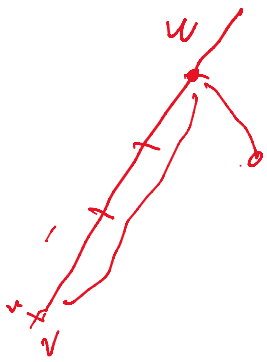
```
    // Dung Centroid Tree
    Pa[r] = dad;

    // Xoay cay
    xoa[r] = 1;
    for (int u : adj[r])
        if (!xoa[u]) {
            Centroid_decompose (u, r);
        }
}

void Xuly0() {
    int v;
    cin >> v;
    color[v] = 1 - color[v];
    if (color[v] == 1) {
        int u = v;
        while (u) {
            se[u].insert (distance (u, v));
            u = Pa[u];
        }
    } else {
        int u = v;
        while (u) {
            se[u].erase (se[u].find (distance (u, v)));
            u = Pa[u];
        }
    }
}

void Xuly1() {
    int v;
    cin >> v;
    int res = n + 1, u = v;
    while (u) {
        if (!se[u].empty())
            res = min (res, *se[u].begin() + distance (v, u));
        u = Pa[u];
    }
    if (res == n + 1)
        cout << -1;
    else
        cout << res;
    cout << '\n';
}

#define task "CCOLOR"
int main() {
```



```
    if (fopen(task".inp","r")) {
        freopen(task".inp","r",stdin);
        freopen(task".out","w",stdout);
    }
    ios::sync_with_stdio (0);
    cin.tie (0);
    cout.tie (0);

    int Q;
    cin >> n;
    for (int i = 1; i < n; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
        adj[v].push_back (u);
    }

    // Xu ly bai toan LCA
    DFS (1, 0);
    BuildRMQ();

    Centroid_decompose (1, 0);

    cin >> Q;
    while (Q--) {
        int loai;
        cin >> loai;
        if (loai == 1)
            Xuly0();
        else
            Xuly1();
    }

}
```